SPRING BEAN DEFINITION

The objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container, for example, in the form of XML <bean/> definitions which you have already seen in previous chapters.

The bean definition contains the information called **configuration metadata** which is needed for the container to know the followings:

- How to create a bean
- Bean's lifecycle details
- Bean's dependencies

All the above configuration metadata translates into a set of the following properties that make up each bean definition.

Properties	Description
class	This attribute is mandatory and specify the bean class to be used to create the bean.
name	This attribute specifies the bean identifier uniquely. In XML-based configuration metadata, you use the id and/or name attributes to specify the bean identifier(s).
scope	This attribute specifies the scope of the objects created from a particular bean definition and it will be discussed in bean scopes chapter.
constructor-arg	This is used to inject the dependencies and will be discussed in next chapters.
properties	This is used to inject the dependencies and will be discussed in next chapters.
autowiring mode	This is used to inject the dependencies and will be discussed in next chapters.
lazy-initialization mode	A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at startup.
initialization method	A callback to be called just after all necessary properties on the bean have been set by the container. It will be discussed in bean life cycle chapter.
destruction method	A callback to be used when the container containing the bean is destroyed. It will be discussed in bean life cycle chapter.

Spring Configuration Metadata

Spring IoC container is totally decoupled from the format in which this configuration metadata is actually written. There are following three important methods to provide configuration metadata to the Spring Container:

1. XML based configuration file.
2. Annotation-based configuration

3. Java-based configuration

You already have seen how XML based configuration metadata provided to the container, but let us see another sample of XML based configuration file with different bean definitions including lazy initialization, initialization method and destruction method:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <!-- A simple bean definition -->
   <bean  >
       <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- A bean definition with lazy init set on -->
   <bean  >
       <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- A bean definition with initialization method -->
   <bean  >
       <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- A bean definition with destruction method -->
   <bean  >
       <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- more bean definitions go here -->

</beans>
```

You can check Spring Hello World Example to understand how to define, configure and create Spring Beans.

I will discuss about Annotation Based Configuration in a separate chapter. I kept it intentionally in a separate chapter because I want you to grasp few other important Spring concepts before you start programming width Spring Dependency Injection with Annotations.